Report Number: WUCS-00-26

2000-01-01

On Maintaining Group Membership Data in Ad Hoc Networks

Gruia-Catalin Roman, Qingfeng Huang, and Ali Hazemi

The design of ad hoc mobile applications often requires the availability of a consistent view of the application state among the participating hosts. Essential to constructing a consistent view is the ability to know what hosts are within proximity of each other, i.e., form a group in support of the particular application. In this paper we propose an algorithm that allows hosts within communication range to maintain a consistent view of the group membership despite movement and frequent disconnections. The novel features of this algorithm are its reliance on location information and a conservative notion of logical connectivity that... Read complete abstract on page 2.

On Maintaining Group Membership Data in Ad Hoc Networks

Gruia-Catalin Roman, Qingfeng Huang, and Ali Hazemi

Complete Abstract:

The design of ad hoc mobile applications often requires the availability of a consistent view of the application state among the participating hosts. Essential to constructing a consistent view is the ability to know what hosts are within proximity of each other, i.e., form a group in support of the particular application. In this paper we propose an algorithm that allows hosts within communication range to maintain a consistent view of the group membership despite movement and frequent disconnections. The novel features of this algorithm are its reliance on location information and a conservative notion of logical connectivity that creates the illusion of announced disconnection. Analysis of movement patterns and delays is used to anticipate physical disconnections before they can impact application results.

On Maintaining Group Membership Data in Ad Hoc Networks

Gruia-Catalin Roman, Qingfeng Huang and Ali Hazemi

WUCS-00-26

April 2000

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

On Maintaining Group Membership Data in Ad Hoc Networks

Gruia-Catalin Roman, Qingfeng Huang, Ali Hazemi
Department of Computer Science
Washington University in St. Louis
St. Louis, MO 63130
Phone: 314-935-6132, Fax: 314-935-7302
{roman,qingfeng,hazemi}@cs.wustl.edu

April 16, 2000

Abstract

The design of ad hoc mobile applications often requires the availability of a consistent view of the application state among the participating hosts. Essential to constructing a consistent view is the ability to know what hosts are within proximity of each other, i.e., form a group in support of the particular application. In this paper we propose an algorithm that allows hosts within communication range to maintain a consistent view of the group membership despite movement and frequent disconnections. The novel features of this algorithm are its reliance on location information and a conservative notion of logical connectivity that creates the illusion of announced disconnection. Analysis of movement patterns and delays is used to anticipate physical disconnections before they can impact application results.

1 Introduction

Ad hoc mobile networks define a new computing environment that consists of hosts traveling through physical space and communicating in an opportunistic manner via wireless connectivity. In the absence of a fixed network infrastructure, the mobile hosts must discover each other's presence and establish communication patterns dynamically. In some cases the hosts share a single broadcast medium in a limited region of space while in other situations they may act as routers for each other thus extending the range of communication beyond the direct reach of the wireless transmitters. Almost always, communication is assumed to be symmetric even though physically it is easy to envision circumstances in which some hosts may be able to reach much further than some others.

The absence of a fixed network infrastructure, frequent and unpredictable disconnections, bandwidth limitations, and power considerations render the development of ad hoc mobile applications a very challenging undertaking. Yet, ad hoc networks are emerging as an important platform for new applications of practical importance. Some applications, such as emergency response to a major disaster, must function in situations characterized by a total collapse of the wired infrastructure while others, such as mine surveying benefit from the flexibility of the ad hoc network structure [2]. Even when the wired infrastructure is available, as the number of devices present in a single rooms grows into the tens or even hundreds it becomes infeasible to provide them all with wired connectivity.

Efforts are under way to construct the kind of infrastructure required to support such applications. Standards, such as IEEE 802.11, IEEE 802.15, and Bluetooth [13], are being developed to make basic wireless connectivity possible. Protocols are being adapted or redesigned to accommodate the characteristics of wireless communication, to facilitate interoperability with the wired networks, to provide ad hoc routing capabilities [1, 9], and to offer broadcast and multicast functionality. More recently, middleware is being developed for coordination and communication in mobile systems, in an effort to make the mobility totally transparent to the application programmer. Typical examples include IBM TSpaces [6], Sun Jini [8], and Lime [10]. TSpaces provides tuple space access and event notification capabilities in a mostly wired environment. Similar reliance on the client-server architecture is found in Jini, which offers support for service registration and discovery. Finally, Lime assumes a coordination perspective that enables application programmers to reduce the effects of mobility to atomic changes in the contents of a virtual global data structure. Its content is dynamically determined by the connectivity among hosts and is presented to the application as a tuple space resulting from the union of the tuple spaces residing on hosts in

proximity to each other. All the three examples pursue the same tuple space communication strategy as in Linda model [5]. This is not at all surprising given Linda's focus on contents based access and spatial and temporal decoupling among processes.

It is Lime that provided the initial impetus for this research. In order to construct transparently a global data structure that is dynamically restructured based on the arrival and departure of neighboring hosts, one must be able to determine who is around at any point in time. This is not an entirely new problem. Discovery protocols are routinely used in establishing connectivity among wireless devices. Group membership is useful in application-specific multicast and in organizing ad hoc networks for efficient message routing. The new element that we bring to this problem is the concern with consistency. From an application perspective, it is desirable in some cases that the data structures presented to the application appear to be the same on all participating hosts, i.e., all operations on the data structures are serializable whether they originate with the application or they are induced by the movement of hosts. A precondition to accomplishing this is the need to maintain a consistent view of the group membership across all the hosts in the group. In this paper we discuss an algorithm for accomplishing this task in an ad hoc environment. We rely on location information to decide when a host within communication range is admitted to or eliminated from a group. The policy is conservative in nature in order to ensure that all communication activities within a group and the changes in group membership appear to be atomic, i.e., serializable transactions. The algorithm is able to accommodate both the merging of groups and the partition of one group into multiple disjoint groups.

The remainder of the paper is organized as follows. Section 2 provides a formal definition of the problem. Section 3 explains the basic idea behind the group membership algorithm considering the special case of a single node joining and leaving an otherwise stable group. Section 4 presents the algorithm. Section 5 analyzes the feasibility of the algorithm. Discussion and conclusions appear in Sections 6 and 7.

2 Problem Definition

In this section we provide a formal characterization of the group membership maintenance problem. We model a mobile ad hoc network as a graph $C_0 = G(V, E_0)$, where V is the set of mobile hosts and E_0 is a set of bi-directional communication links among the hosts. Graph C_0 changes over time. The presence of an edge (u, v) indicates that host u is within transmission range of host v, and vice versa. We refer to this graph as the physical connectivity graph. In practice, each host can make itself known to its neighbors by generating a beacon at regular intervals and by listening to signals from other hosts around. When a beacon ceases to be heard, a node is considered to be no longer within transmission range. The frequency of the beacon transmissions determines the accuracy of the information available at each host.

Since any attempt to maintain an accurate picture of the physical connectivity graph C_0 in the presence of unexpected disconnections is infeasible, we introduce the notion of a logical connectivity graph, call it $C = G(V, E)$. The latter is a subgraph of the former. The two share the same set of vertices (hosts) but the logical graph is missing some of the edges (links). The choice of edges to include in the logical graph is determined by some group management policy designed to overcome the difficulties caused by unannounced disconnections taking place in the physical system. The choice of policy is not relevant at this point in the definition but it is critical when it comes to demonstrating the feasibility of the algorithm used to solve the group membership maintenance problem. Before proposing our formulation of the problem we need to define one more concept, the notion of a group. A group G is simply a maximum-sized connected subgraph of the logical connectivity graph C. Since each host u is always a member of some group, we use $G(u)$ to denote the group that includes u and we extend the notation to $V(u)$ and $E(u)$ to refer to the vertices and edges of the group. Clearly, the logical connectivity graph C is always partitioned into a set of disconnected groups. As the underlying physical connectivity graph changes so does the logical one. The group management policy is assumed to add a new edge to the logical graph after it appears at the physical level and to remove it from the logical connectivity graph before it is likely that it might disappear from the physical graph.

The group membership maintenance problem is defined as the requirement for each host in the logical connectivity graph to have knowledge of what other hosts are members of its group and for such knowledge to be consistent across the entire group at all times. Any feasible solution to this problem requires one to make some reasonable assumptions about node movement and network delays, to define a group membership policy and to develop a protocol that serialize all configuration changes. The basic ideas behind our solution are explained in the

next section by considering a special case when a single host is added and deleted from an otherwise stable group.

3 Solution Idea

In this section we introduce the key ideas behind our solution to the group membership maintenance problem. We do so by considering the special case of a single host u joining and leaving a group G in an ad hoc mobile network. The group is assumed to have a leader whose main function is that of serializing all configuration changes. In our special case, it is the leader that decides when a host is admitted to or eliminated from the group, and make each member know the composition of the group. The group leader informs all the members about changes to group membership thus guaranteeing that members have the same view of the group. The leader's decisions are based upon the relative locations of all the hosts within communication range with the group.

When an isolated host u discovers a neighboring host v that is already in the group G, it asks v about the identity of the group leader. Here we assume that hosts are aware of other hosts within their transmission range. This is realized by having each host transmit a beacon at regular intervals and by listening to signals from other hosts around. We also assume that host failures do not occur and that communication channels between hosts are reliable, bidirectional and FIFO. Under these assumptions, we are guaranteed that host v will receive this inquiry. Once u determines the identity of the group leader l, direct message communication with the group leader becomes possible and host u will start reporting to the group leader its position at regular intervals. We assume that all members of the group do the same. Here we make the additional assumption that ad hoc routing is available in the network. Once the group leader l has decided to admit u to the group, it informs u and all the members of the group about the configuration change by sending them a message called the *join message*. The group leader makes its decision based on the relative locations of host u and of hosts within u's communication range. When a host receives a *join message* from the leader, it stops transmitting regular messages and sends a *flush message* to all the other hosts within the group indicating it knows about the configuration change. Once a host receives a *join* message from the leader and *flush messages* from all the members of the group, it adds the new host u to its list of group members, i.e., commits to the new group configuration which includes host u as a member, and resumes sending regular messages. The leader will also wait to receive *flush messages* from all the members of the group before updating it's group membership list. Since we are assuming FIFO communication channels, receiving the *flush message* on a particular link guarantees that there are no more messages in transit on that link that may have been sent in a prior group configuration. Mobile host u has now successfully joined the group G and all the members of the group have the same view of the group, that is, they have the same group membership list. An illustration of host u joining group G is depicted in Figure 1.

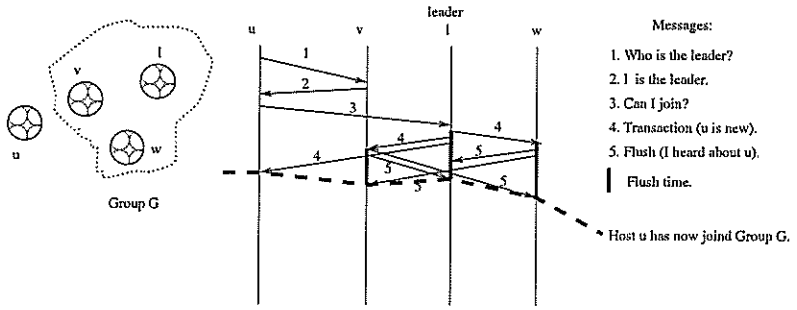

Figure 1: Host u joining group G

Let us now describe how a host w leaves its group G. When the leader decides that w can no longer be part of the group, it sends a *depart message* to all the members of the group. All group members process the *depart message* similarly to processing a *join message* and, once they receive all the *flush messages* from other group members, they remove w from their group membership list. Mobile host w is no longer part of the group G.

Why do we need the *flush messages* in the algorithm? They are needed in order to make the group configuration changes appear to be atomic. To examine this notion further, consider the scenario where hosts v and w belong to a group G whose leader is host l. Let's also assume that l decided to have host u join the group and sent a *join message* to all the hosts in the group. Moreover, imagine that v just sent a message M to w before receiving

the *join message* from the leader and the message has not arrived at w yet. The *join message* arrives at w, which swiftly updates its membership list, without waiting to receive any *flush messages*. At a later time message M finally arrives at host w. Notice that when message M was sent, the group membership list contained hosts l, v and w and when the message is received the membership list also includes u! In other words message M was sent in one group configuration and received in another. While we desire to make group configuration changes atomic, in the above example message M crosses the transaction in which u joins the group G. However by waiting for all *flush messages*, all communications between hosts within a group take place either before or after u joins the group, i.e., the configuration change appears to be atomic. This situation is depicted in Figure 2.

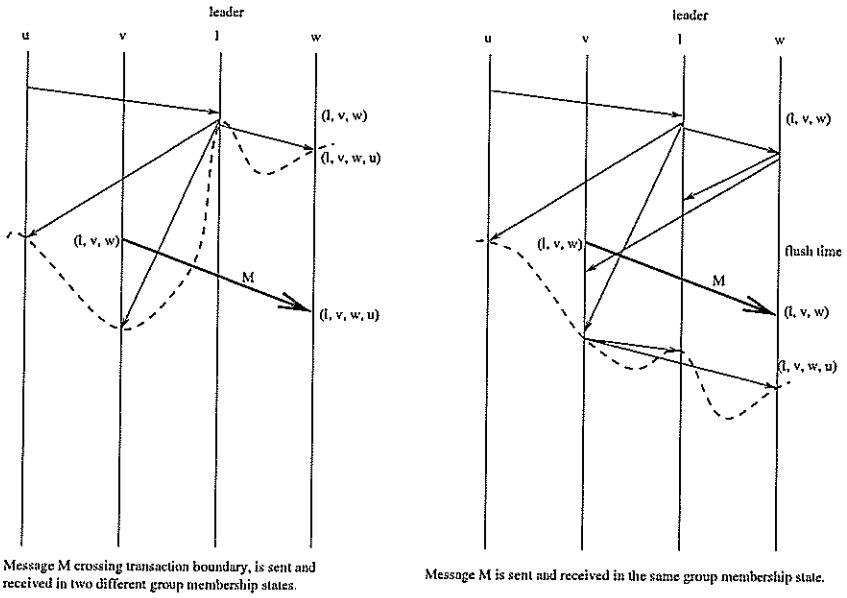

Figure 2: Message crossing the transaction boundary

Let us now discuss the group membership policy we are proposing in our solution. A host is admitted into the group, only if we can guarantee reliable message delivery to that host from any other host present in the group. Let us first discuss how we can provide guarantees of message delivery between two mobile hosts, we then describe the more general case of how this guarantee is provided within a group. Consider a host u with transmission range R. In absence of mobility any host v within this transmission range is guaranteed to receive messages sent by u. However in presence of mobility, we can not provide this guarantee since v may not stay within u's transmission range to receive messages sent by u. To overcome this difficulty, we logically restrict the transmission range of host u to a smaller area r, where we can provide the message delivery guarantee, call it the safe transmission zone of host u. If host v is within the safe transmission zone of u, only then we consider v connected, i.e. able to communicate, with u. By restricting host u's range, we are able to warp up any communication between u and v before v moves out of transmission range of u. To determine r, we make the conservative assumption that hosts u and v are moving away from each other at a velocity which is the larger of their two velocities, $Vmax$, and it takes $Ttrans$ time to complete a transaction between the two hosts, i.e., the round trip message passing delay between the two hosts has the upper bound $Ttran$. The displacement between u and v within $Ttran$ time is no more than $((2 * Vmax) * Ttrans)$, hence $r = R - ((2 * Vmax) * Ttrans)$. We can now guarantee that any messages sent from host u to $v1$, a host located within u's safe zone, will be received safely because $v1$ remains within u's transmission range by the time a message arrives at $v1$. An illustration of the safe zone of host u is depicted in Figure 3.

When a host u joins a group, we must consider a displacement of $((2 * Vmax) * Tnetwork)$, where $Tnetwork$ is the round trip message passing delay within the group, to determine the safe zone of u's neighboring hosts. If host u is within the safe zone of any neighbor, then u is admitted to the group since we can provide message delivery guarantees from any host in the group to u. This will allow us to provide guarantees of message delivery from any host present within the group to u.

In the following section we present the algorithm in its entirety.

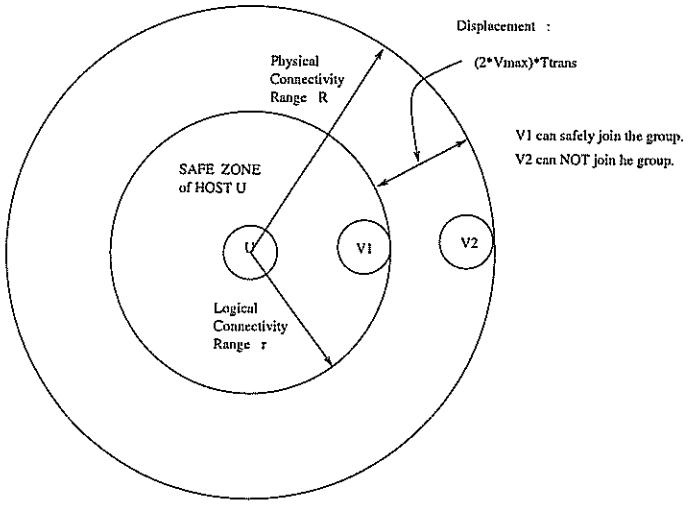

Figure 3: Safe transmission zone of host u

4 Algorithm Description

In the previous section we described the main idea of the algorithm using the example of a single node joining and leaving a group. In general, whole groups can be merged and split in arbitrary ways. In this section we describe the general algorithm. When a mobile node first comes to life, it declares itself the leader of a group containing itself as sole member. At this point, it begins to run a neighbor-discovery protocol (Section 4.1) designed to find if there are any other groups around. When a new group is found in the vicinity, the two groups may negotiate a merger (Section 4.3). The conditions for the merger may consider issues such as the guarantee that no physical partitioning is likely to take place within a certain amount of time. The merging protocol uses a leader election scheme which ensures that only one leader is associated with the new group, i.e., the other nodes become regular members. When a group contains more than one member, it is possible for some members of the group to move out communication range relative to the others in the group. If this happens in an unpredictable way (unannounced disconnection), it is impossible to maintain a consistent view of the group membership. To prevent unannounced disconnection, the leader seeks to anticipate possible partitions (Section 4.4) by re-examining the overall configuration anytime a member changes location. When the leader expects a physical partition of the group to take place, it forces an announced disconnection (Section 4.5), which guarantees that a logical disconnection occurs before any physical disconnection becomes possible. Predicting partitions is made possible by the fact that members of a group constantly update their leader with regard to their current location (Section 4.2).

We assume that each mobile node has a unique identifier (id). We choose the identifier to be the network address of the node. We also assume that each participating mobile node is aware of its own location (xy), which could be acquired from a GPS system, for example. A node is always a member of a group, which is identified by a group identifier (gid). We choose the group identifier to be the leader's id. Each node also has a view (π) of its group, i.e., an (up to date) list of group member identifiers. In order to serialize all configuration (view) changes, every member of a group keeps track of a group transaction number (τ). The group leader also records members' last reported location in a group map (Π). To support neighbor discovery, each node keeps a list of recently detected neighboring groups (ξ). Given the information gathered during neighbor discovery, the group leader generates a list of merging contacts (Θ). The leader only sends merger requests to groups represented in Θ. All these variables used in the protocol are summarized in the table.

State	
id	:node identifier, contains network address
xy	:node location, constantly updated by external mechanism
gid	:group identifier, contains leader's network address
τ	:group transaction sequence number
π	:group member list;
ξ	:the list of newly discovered neighboring groups
Π	:group map, not null only in leader state, includes all member locations
Θ	:the list of merging contact, not null only in leader state.
Ψ	:partition set, non-empty only in leader state

Next we discuss the protocols in detail.

4.1 Neighbor Discovery

The neighbor discovery protocol is responsible for detecting the presence of new groups. In our model, all members of a group participate in the discovery process, i.e., each member acts both as an eye and a voice for the group. When a node detects a new neighbor from the information available in the MAC layer, it broadcasts (one-hope multicast) a greeting to all its neighbors. The greeting contains the sender's group identifier (*gid*). When a node receives a greeting, it compares the *gid* in the greeting with its local group identifier and the group identifiers in ξ to see if *gid* is a new group identifier. If it is new, the *gid* is time-stamped and added to ξ; if the node already has a copy in ξ, the time-stamp is updated. The list ξ could be out of date because of mergers, leader changes, and network splitting. For example, a group which did not experience splitting and did not encounter other groups for some time should empty the contents of their ξ. A scavenger process is run to clean up the neighbor list constantly, throwing out old *gid*s. In order to keep other groups constantly aware of its existence, each node periodically broadcasts the neighborhood greeting if its ξ is not empty. By utilizing the neighborhood information available at the MAC layer, the discovery cost is kept small especially when compared against approaches that use their own periodic discovery messages.

4.2 Location Update and Discovery Summary

The location update process is responsible for keeping the leader updated of all its members' current locations. All members periodically send their location information to the leader. Whenever it receives a location update from a member u, the leader updates the group map Π by replacing u's old location with the new one. The discovery summary process is responsible for finding out how many other groups are in one's vicinity. All members are required to send their individually discovered neighborhood information ξ to the leader. By putting the discovery information together, and by comparing it with its current contact list Θ, the leader gets to know what other groups are currently connected with its group. This updates the contents of the contact list Θ. Considering the fact that the location information only uses a few bytes, and ξ normally has only a few bytes as well, we piggyback ξ on the location update message. This way we reduce some of the communication overhead.

4.3 Merger

The merger process uses a three step scheme to negotiate and carry out the task of merging two or more groups into one. A leader periodically proposes a merger to the groups in the vicinity by sending a merger request to the contact(s) in Θ. The merger request includes sender's group map Π for the recipient to decide whether to accept the request or not. When a leader l receives a merger request, it sends back a NACK if it is already engaged in a merger (e.g., just acked a merger request) or it finds that it's not safe to merge with the sender's group according to policy P and the maps of both groups. Otherwise, leader l sends back an ACK with the information of its group identifier *gid*, group roster π, and group transaction number τ. When the initiating leader receives all the expected ACK(s) and NACK(s), it issues a merging order to all the members of its own group and to all members of the groups that agreed to merge. The merging order contains the complete information about the new combined group. When a node receives a merging order, it first sends a flush message to all the current members , stops sending any regular messages and waits until it received all the expected flush messages from all the current group

members. At this point it is safe to change the view [1], it resets its *gid*, view π, and transaction number τ using the information in the merging order. For the leader, it also resets the contact list Θ after the merger is finished (Θ will eventually be replenished by new neighborhood discoveries.).

What if some contact information in Θ is no longer valid at the point when the leader gets to send out the merger request? This is possible because, for example, a group p was discovered by group q in its vicinity, but p quickly moved away before the leader of q gets to send a merging request, or p's leader may have changed and the request is received by a node that is no longer a leader. In the first case, the network returns a network exception, the sender knows the group is out of contact. In the second case, the receiving node is obligated to send back a NACK. [2] For convenience, we treat the network exception as a NACK in our discussions.

Once a leader has sent out the merger request(s) (proposing a merger), it is obligated not to issue any other transaction until the merger status is clear, that is, either the merger is finished, or the merger is canceled. The proposed merger is canceled if all the expected groups sent negative replies. After a leader acknowledged a merger request, it surrenders its leadership status and becomes a regular node. As a regular node, it will reject any further merger requests. Of course, regular nodes may receive merger request(s) because some leader maybe working with an outdated contact list.

4.4 Partition Anticipation

Partition anticipation is the key to preventing mobility-induced unannounced disconnection. Whenever the group map is updated, the leader checks the new configuration to see if the group is in danger of being physically split soon[3]. If there is a possible member location change which might split the physical connectivity graph underlying the group in time less than $T = t_0 + t_r + t_d$[4] but greater than t_0, the algorithm produces a partition scheme Ψ. Where t_0 is reserved time for possible partitioning process, and Ψ is used by the leader in the partitioning process (Section 4.5).

Currently we are using a very simple partition anticipation algorithm. By assuming that each mobile node has a maximum speed V_{max}, each node can move at most $(V_{max}T)$ distance away from its current location in T units of time. The maximum relative distance change between two nodes is $(2V_{max}T)$ during T. It is safe to say that two nodes which are less than $(R - 2V_{max}T)$ distance apart will stay physically connected for at least T more units of time, where R is the maximum communication range between the two nodes. For a group of more than two mobile nodes, we can be sure that the group stays physically connected for at least T units of time if there exists a Spanning-Tree in the geometric connectivity graph [5] such that all the tree edges have length less than or equal to the safe distance $r = R - 2V_{max}T$. By putting edges only between nodes which are less than r apart, we get a graph $G_r(V, E)$ for the group. Using depth-first search, we can determine how many connected subgraphs are in $G_r(V, E)$ in $O(N)$ time [6]. Each connected subgraph is a T-safe configuration, i.e, the nodes in the subgraph are guaranteed to stay physically connected for at least T units of time.

4.5 Partitioning

The partitioning process is responsible for splitting a group into multiple groups. When a group leader anticipates a physical partition of the group, it immediately issues a partition transaction order to the members. Each partition transaction order is destination specific, i.e., it contains all the information regarding the new designated leader and new group view for each recipient. When a node receives a partition transaction order, as in the case of mergers, it again first sends out a flush message to each group member, stops sending any more regular messages (receiving is fine), waits until its safe to change the group configuration (i.e.,when all expected flush messages for the partition

[1] We require all regular messages are delivered in the same view, i.e, corresponding sending and receiving events must be in the same view. If the network has a small delay bound t_b, we could choose to let each node wait for t_b time after receiving a merging order, before changing group information.

[2] It is ok if the node has become the leader of another group.

[3] We use worst case analysis, as we want to provide absolute guarantee.

[4] Assuming that the members report their location in every t_r time units, the network delay upper bound is t_d, and the maximum time it takes for a stable group membership being established after a configuration is computed, then the maximum time gap between two consecutive checking of group configuration is $t_r + t_d$. (e.g. the first report arrived immediately, the second report enjoyed the max delay (Figure 4)).

[5] Geometric connectivity graph defined as the physical connectivity graph with each node has location information.

[6] In our analysis we will assume the processing time is sufficiently small to be treated as a constant

are received), then resets the leader identifier and group view using the information in the partition order. The flushing ensures that no regular message is sent and received in different group views.

5 Algorithm Analysis

The key feature of our algorithm is the use of location information in the group membership management. The leader of a group frequently checks the members' location to make sure that only those that are guaranteed to stay connected with the group for at least T more units of time remain in the group. A safe distance r, determined by $(r = R - 2V_{max}T))$ is used to decide if a node should be admitted or deleted from the group. How to determine T? Let's assume t_d is maximum delay time between a control message being issued and being received and processed, which is the sum of the maximum network delay and and the maximum process queuing delays both at the sender and the repceipient sides. For the convenience of discussion, we refer t_d as the network delay. In the case of splitting, the maximum time it takes for a group to be partitioned successfully is twice the network delay. (when both partition order and flush messages enjoyed maximum delay). If the leader continuously monitors group configuration and all the member location were up to date, all possible mobility-induced unannouced disconnection would be caught in advance and dealt with successfully by requiring $T > 2 * t_d$. Yet, the leader's information about members location is always a little bit out of date. If the members sample and report their location every t_u units of time, then the location information the leader has about a member could be outdated by time $t_u + t_d$. Taking this into consideration, the reserved time T must be greater than $t_u + t_d + 2 * t_d = t_u + 3t_d$. Yet, again, $T > t_u + 3t_d$ is not enough, because the leader has to guarantee the new groups are safe right after the splitting. So T must be no less than $2(t_u + 3t_d)$. In the case of merging, the leader should also take into account one trip delay for the merging acknowledgement, T must be no less than $2t_u + 7t_d$, In other words, if the safe distance r used in the partition anticipation should be

$$r = R - 2V_{max}(2t_u + 7t_d)$$

Our algorithm also reqires V_{max} to be less than $V_{admissible}$, where $V_{admissible}$ is the maximum admissible speed for the specific wireless network systems the mobile hosts are using. Most wireless network systems, like DECT, GSM, PCS, ETACS, have a maximum addmissible. When a mobile node is moving too fast, it simply becomes invisible to the network. For GSM and PCS systems $V_{admissible}$ is about $50m/s$, for DECT microcellular system $V_{admissible}$ is about $11m/s$.Without the condition of V_{max} being less than $V_{admissible}$, a speed change from $V < V_{admissible}$ to $V > V_{admissible}$ will create an unannouced disconnection too, thus we would have to add speed monitoring besides the location monitoring to prevent this speed change induced unannouced disconnection from happening.

Figure 4 is illustrates the relation between the safe distance r and the maximum network delay t_d with reasonable values of $R = 500meters$, $V_{max} = 20meter/second$ and location report frequency of 1 Hz $(t_u = 1s)$. It shows the intuitive result that when a network has higher delay bound, the safe distance used in partition anticipation shall smaller.

Figure 5 shows the relation between safe distance, speed upper bound and network delay bound. The region above the top curve corresponds to $r < 0$. It means that given the node speed upper bound and network delay bound in that region, we could not provide any group view consistency guarantee for a group of more than one member. The region below $r = 300m$ corresponds to $r > 300m$. It means that if a mobile system's network delay bound and maximum speed bound fall into the region below the $(r = 300m)$ curve, we could provide the group view consistency guarantee by using $r = 300$ in partition anticipation.

The correctness of our algorithm also relies on the assumption that the network has a delay bound. At time moment, we are not aware of any ad hoc routing protocols which can provide a good delay bound. Yet, it is conceivable that a routing protocol with good delay bound for prioritized group control messages is possible by restricting group size and using location information. We are currently exploring in this direction.

6 Discussion

Maintaining a consistent view of the global state in a distributed network is difficult in general and almost impossible in presence of unannounced disconnection. In ad hoc mobile systems, mobility-induced unannounced disconnection is a frequent event, part of the normal operation of the network. In this paper, we have presented an algorithm that maintains a consistent group membership view in ad hoc network. The novel feature of this algorithm is its

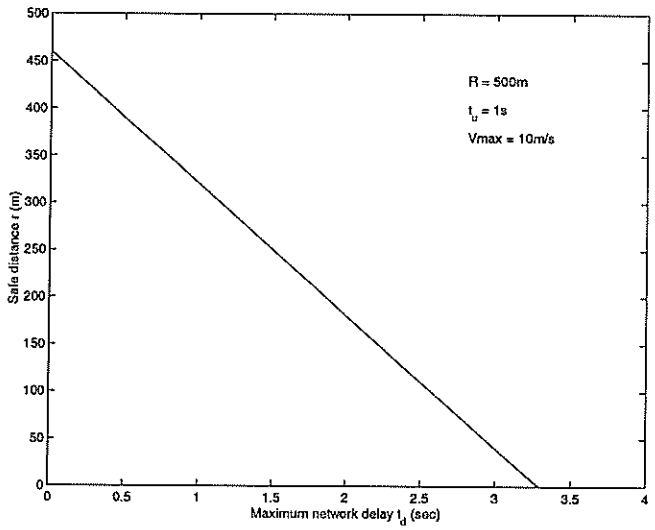

Figure 4: Safe distance v.s network delay

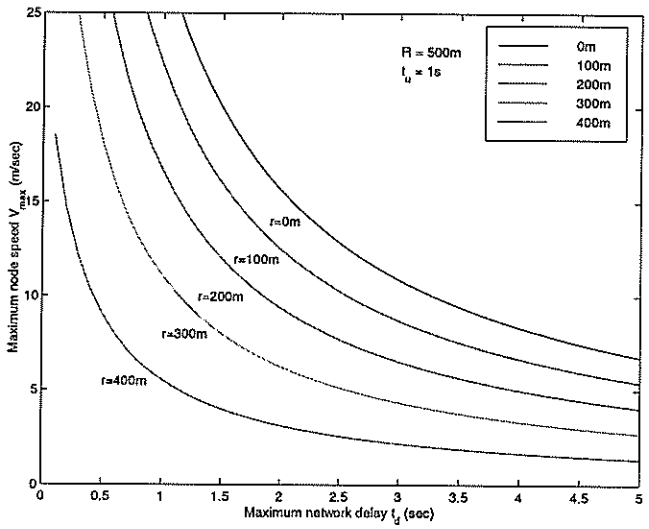

Figure 5: Relation between safe distance, speed bound and delay bound

ability to create the illusion of announced disconnection. By using location information for the mobile hosts in the region, the membership service is able to guarantee to the application layers results that are not affected by mobility-induced unannounced disconnection.

The idea of using location information in ad hoc network is not entirely new. Ko and Vaidia, for instance, used location information to improve the efficiency of ad hoc routing [7], Prakash and Baldoni used location information in the determination of group membership assuming the network stays connected [11]. The group membership service were traditionally studied [4] for distributed applications running on top of a reliable, usually fixed, wire-line network in which link failures and network partitions are rare. The Transis project [3, 12] has dealt with membership services and group communication in environments in which the network itself may get partitioned due to node and link failures, and nodes may operate for extended periods of time in disconnected mode.

At present, our algorithm uses the assumption that the mobile nodes in the system have a known maximum speed. Unbounded speed range is another possible source of unannounced disconnection due to the low speed requirement for wireless networks. In the systems that don't have a small speed bound, i.e. , smaller than $V_{admissible}$, but the mobile nodes can determine their own speed or velocity, like most cars and planes do, a safe

velocity threshold can be used in the decision of merging and splitting. Of course, in such cases we would have to assume a maximum acceleration for the mobile nodes. The idea of whether and how speed and velocity information can be used to improve the efficiency of the algorithm needs further investigation. The algorithm could also be enhanced by considering security issues , or some QoS factor. Besides all these, an immediate issue on our research agenda is finding an ad hoc routing protocol which can provide a small delay bound for group control messages, and/or comparing existing protocols on the issue of delay by simulation.

7 Conclusions

The principal motivation for this work was desire to provide consistencies to applications that execute over ad hoc networks. The idea is to allow them to view and modify data that is distributed across mobile hosts as if it were accessible through global memory. The first step in this direction was to develop the ability to maintain a consistent view of the list of participants in the application. Despite the presence of disconnections, we were able to accomplish this by drawing a sharp distinction between physical and logical connectivity, i.e., by having members be admitted and eliminated from a group only when the operations are guaranteed to succeed. The approach is representative of a new direction in networking, one that factors into the protocols information about mobility and space.

Acknowledgments: This research was supported in part by the National Science Foundation under Grant No. CCR-9970939. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The authors wish to thank Amy Murphy for helpful discussions during the development of these algorithms.

References

[1] J. Broch, D.B. Johnson, and D.A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks . Internet Draft, October 1999. IETF Mobile Ad Hoc Networking Working Group.

[2] R. Brooks and J. McLurkin. Using Cooperative Robots for Explosive Ordnance Disposal. http://www.ai.mit.edu/projects/microrobots/. Massachusetts Institute of Technology Artificial Intelligence Laboratory.

[3] Malki D. Dolev, D and R. Strong. A Framework for Partitionable Membership Service. Technical Report CS95-4. The Hebrew University of Jerusalem.

[4] R. Friedman and R. van Renesse. Strong and Weak Virtual Synchrony in Horus. Tr95-1537, Cornel University, Department of Computer Science, 1995.

[5] D. Gelernter. Generative Communication in Linda. *ACM Computing Surveys*, 7(1):80–112, Jan. 1985.

[6] IBM. T Spaces Web page. http://www.almaden.ibm.com/cs/TSpaces, 2000.

[7] Y.B. Ko and N.H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proc. ACM/IEEE MOBICOM '98*, October 1998.

[8] Sun Microsystems. Jini web page. http://www.sun.com/jini.

[9] C.E. Perkins, E.M. Royer, and S.R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. Internet Draft, October 1999. IETF Mobile Ad Hoc Networking Working Group.

[10] G.P. Picco, A.L. Murphy, and G.-C. Roman. LIME: Linda Meets Mobility. In D. Garlan, editor, *Proc. of the 21st Int. Conf. on Software Engineering*, pages 368–377, May 1999.

[11] R. Prakash and R. Baldoni. Architecture for Group Communication in Mobile Systems. In *Proc. of the IEEE Symposium on Reliable Distributed Systems (SRDS)*, page 235, October 1998.

[12] K.P. Birman R. van Renesse and S. Maffeis. Horus, a flexible Group Communication System. *Communications of the ACM*, April 1996.

[13] Bluetooth SIG. Bluetooth web page. http://www.bluetooth.com.